Embedded Software Update Methods

[001] **Related Applications**

[002] The present application claims priority from a U.S. provisional application, Application No. 60/416,697, entitled "Embedded Software Update System", filed on October 8, 2002. The prior application is hereby incorporated into this application by reference as if fully set forth herein. The present application is also related to a PCT application, filed July 15, 2002, with International Application No. PCT/US02/22412 and International Publication No. WO03/00913 A1, entitled "EMBEDDED SOFTWARE UPDATE SYSTEM." The PCT application is also incorporated into this application by reference as if fully set forth herein

[003] **Field of the Invention**

The present invention is related to digital electronics products with embedded software operating systems, and more particularly related to methods of updating, correcting, modifying or upgrading the embedded software in such digital products before or after the products have been released to market.

[004] **Art Background**

[005] For the digital products, such as cellular phones, PDAs and set-top boxes, their digital technology is based on large-scale embedded software system. Despite improvement in various stages of digital product development and manufacturing, after a digital product has been released to the field, manufactures will sometimes go through improvement, enhancement or upgrades, which may require a change in the embedded software.

[006] Software update systems for embedded software have been disclosed. However, they have not provided an efficient way for updating embedded software with small amount of update data. They also have not taught how to transmit such software update data to the digital products.

[007] **Summary of the Present Invention**

[008] The present invention is directed to a method of utilizing text messages for

transmitting software update data or software patches for updating embedded software in the digital products. Based on this method, software update data or software patch data can be transformed into text messages and transmitted to digital products using text message services, such as Short Message Services (SMS) in cellular phone system. By the phrase "text messages" it is broadly meant a string, or more, of text characters.

[009] A further aspect of the present invention introduces a method of updating embedded software in the unit of software functions. This approach can separately update one or more software functions without changing program code in the other functions.

[010] **Brief Description of the Drawings**

[011] Fig. 1 illustrates a simplified diagram of an exemplary process of preparing software patches as a string of text characters for transmission and an exemplary process of restoring patch data from received string of text characters in accordance with one embodiment of the present invention.

[012] Fig. 2 illustrates exemplary embedded software composed by regular software functions.

[013] Fig. 3 illustrates exemplary embedded software composed by software functions and Header-Areas.

[014] **Detailed Description of the Preferred Embodiment**

[015] 1. A method of utilizing text messages, or string(s) of text characters, for transmitting software update data or software patches to update embedded software in the digital products is disclosed. Another method of separately updating one or multiple software functions without changing software code in the other functions also disclosed. In the following detailed description, numerous specific details are set forth to provide a full understanding of the present invention. It will be obvious, however, to one ordinarily skilled in the art that the present invention may be practiced without some of these specific details. In other instances, well-known structures and techniques have not been shown in detail so as to avoid unnecessarily obscuring the present invention. Additionally, headings are used throughout the description as a matter of convenience and for ease of references only. It should note that what is meant by "embedded software" herein is the software program written in C/C++/Java/Assembler, or other software programming languages, as well as any executable machine code of CPU/Microprocessor or DSP ("Digital Signal Processor"), for operating the digital product. . It should also note that what is meant by "software patch" herein is a block of software update data for updating the embedded software in the digital products.

[016] 2. Using text messages to carry software patch data

[017] A software patch or a block of software update data, for updating the embedded software in a digital product, can be expressed by a string of 8-bit bytes, D_1, D_2, D_3, ..., D_N, where N is the number of total bytes. When a plurality of text characters, such as a text message, is used for carrying the update data in a software patch, it is necessary to ensure that every data byte in the update data can be transmitted correctly to the digital product without data change or data loss during the transmission.

[018] As can be appreciated by those skilled in the art, most text characters can be represented by a string of 8-bit characters or a string of 16-bit characters. This invention does not limit usages of any language characters, 8-bit characters or 16-bit characters, but uses the text string of 8-bit characters as an example to show the invention details.

[019] Conventionally, when a string of text characters composed by 8-bit characters, for example ASCII characters, is transmitted through the Internet or wireless networks, some changes may be made on the text characters by the Internet or wireless network. For example, when a text string, composed by 8-bit ASCII characters, is sent as an email message to the Internet, for each 8-bit ASCII character, only 7 bits in a character (7 bits of LSB - Least Significant Bits,) can be correctly transmitted through the Internet; while the data of the one bit of MSB (Most Significant Bit) is lost during transmission. Another fact in text string transmission is that, not every character of ASCII code can be transmitted correctly through Internet or wireless network, which will cause problems when text strings are used to transmit software update data to the digital products.

[020] The first embodiment of the present invention is directed to a method of utilizing text strings for transmitting software update data or software patches. The general idea is to utilize a given set of characters, which can be correctly transmitted through Internet and/or wired or wireless communication links, to contain the information of software update data. In the following, the Short Message Service (SMS) of cellular phone system is used as an example to show this method. Similar methodology can be used in other kinds of text message services in wired or wireless communications. **Figure 1** illustrates a method of using text message to transmit software patch data. As shown in **Figure 1**, this method is composed of three steps. As the first step, before transmitting a software patch to a communication link, a Text-Encoder is used to transform the corresponding patch data into a string of text characters, which composes a text message. As the second step, transmit the text message transformed from the software patch through a communication link to the digital product. As the third step, after the text message is received in the digital product, a Text-Decoder is used to restore the text message back to

the software patch data.

[021] 2.1 Methods of using SMS messages for transmitting software patches

[022] Short Message Service (SMS) has been widely used in cellular phones, wireless PDAs, and some other types of wireless digital products. The "payload" or "user data" of a SMS message can be text sentences or some other types of data type that can be in the unit of 7-bit or 8-bit or 16-bit or some other lengths.

[023] Text messages of SMS can be sent to a SMS service center through Internet as a regular email, or from an Internet web site that supports SMS message input, or from a cellular phone that supports SMS message input. The SMS service center will pass the message contents to cellular phones via SMS messages using a wireless link. However, not every kind of character can be transmitted correctly through the above transmission paths. For example, when a text message is represented by ASCII code, and has characters of values ranged from 0 to 31, there may be problems to transmit the message as a regular text message to the SMS service center. One problem is that some of the ASCII characters ranged from 0 to 31 cannot be directly input into a regular email or into a web site that supports SMS message input, because the characters are not regular text characters. Another problem is that some of the ASCII characters ranged from 0 to 31 can be treated as special control commands instead of regular text characters. For example, when there is a combination of three ASCII characters, i.e., ASCII code 10 <LF>, ASCII code 46 <.>, and ASCII code 10 <LF> in the message which will be transmitted as a regular email, this combination has the meaning of "end of the email" according to the email standard of SMTP (Simple Mail Transfer Protocol, by RFC821). Therefore, the message transmission with email service will be terminated at the place where such a combination appears, and the rest of the message cannot be transmitted.

[024] Therefore, when SMS messages are used to carry software update data for the digital products, it is necessary to avoid the above problems and make sure the entire message content that contains software update information can be transmitted correctly. This invention discloses a method shown in the following.

[025] In order to transform the N bytes of data into text messages, a "Text-Encoder" is introduced. The input of the Text-Encoder is the N bytes of data, and the output of the Text-Encoder is a character string of text message. A set of text characters, named as "Usable-Set" will be selected at first. The Usable-Set is a set of text characters that can be correctly transmitted through all the necessary message transmission paths, such as Internet and/or wireless network links. The Text-Encoder can be designed in such as a way that every output character from the Text-Encoder belongs to the Usable-Set.

Therefore, the text messages generated by the Text-Encoder can be correctly transmitted through all the necessary transmission paths.

[026] For example, if a text message of ASCII characters is used for carrying a software patch, the Usable-Set can be designed as follows. Instead of using the whole range of the ASCII characters that have values from 0 to 127, (though there is the extended ASCII character set for values from 128 to 255, they are not widely used in regular text messages), a set of ASCII characters that have values from 32 to 127 can be considered. However, for the SMS message transmission in the GSM (Global System for Mobile Communication) cellular phone system, some message characters have different values than ASCII character values. For example, ASCII character '@' has value of 64; while the same character '@' has value of 0 (zero) in the GSM SMS message standard. Therefore, when a message written in ASCII characters that contains character '@' is transmitted to a GSM cell phone as a SMS message, the value of '@' will be changed to the value defined by the GSM SMS standard. In order to correctly transmit software update data using SMS messages to GSM cell phones, the characters in the Usable-Set should not contain any characters like '@' that have different values in ASCII standard than the values in GSM standards.

[027] A complete exemplary design on the Usable-Set and the Text-Encoder based on that above analysis is shown as follows. First, a set of ASCII characters that have values from 32 to 95 is selected. Considering the difference between ASCII character definition and GSM character definition, characters that have values of 36, 39, 64, 91, 92, 93, 94 and 95 are replaced by values of 110, 111, 112, 113, 114, 115, 116 and 117 to ensure correct transmission. Therefore, the Usable-Set is the set of characters that have values from 32 to 95, but with several characters replaced with characters that have values from 110 to 117, as described on the above. The process details in the Text-Encoder can be described as follows. The N bytes of software update data or software patch data D_1, D_2, D_3, ..., D_N can be treated as 8N (8 times N) bits of input data string.

[028] Step-1: The Text-Encoder takes 6 bits from the beginning of the input data string and treat they as a 6-bit number. Note that a 6-bit number has a value ranged from 0 to 63.

[029] Step-2: The Text-Encoder adds 32 to the 6-bit number of Step-1. Therefore, the number has a value ranged from 32 to 95.

[030] Step-3: If the number of Step-2 is 36 or 39 or 64 or 91 or 92 or 93 or 94 or 95, the Text-Encoder replace the number by 110 (for 36), or 111 (for 39), or 112 (for 64), or 113 (for 91), or 114 (for 92), or 115 (for 93), or 116 (for 94), or 117 (for 95).

[031] Step-4: The Text-Encoder put the number from Step-3 as a text character into output text message.

[032] Step-5: Go back to Step-1 for the next 6 bits input data, until all the bits in the input data are processed.

[033] In this way, the generated output text messages from the Text-Encoder only use characters in the Usable-Set, and can be correctly transmitted through Internet and wireless network, such as GSM cellular phone network.

[034] On the receiver side of the SMS message, such as a cellular phone that is receiving software update data, the received SMS text messages can be transformed back to the original N bytes of software update data. A "Text-Decoder" is introduced to make such transformation. Following the above exemplary design of the Text-Encoder, the detailed process of an exemplary Text-Decoder can be shown as follows. The received SMS text message can be treated as a string of text characters.

[035] Step-1: The Text-Decoder takes a text character from the beginning of a received SMS message, and treats it as a 7-bit number.

[036] Step-2: If the number of Step-1 is 110 or 111 or 112 or 113 or 114 or 115 or 116 or 117, the Text-Encoder replace the number by 36 (for 110), or 39 (for 111), or 64 (for 112), or 91 (for 113), or 92 (for 114), or 93 (for 115), or 94 (for 116), or 95 (for 117).

[037] Step-3: The Text-Decoder subtracts 32 from the number generated by Step-2. Note that after this step, the number is ranged from 0 to 63.

[038] Step-4: The Text-Decoder treats the number generated by Step-3 as a 6-bit data, and put the 6-bit data into an output data buffer.

[039] Step-5: Go back to Step-1 for the next character in the received message, until all the characters in the received message are processed.

[040] During SMS message transmission and processing, 8-bit text characters may be changed to 7-bit text characters by removing one bit (the Most Significant Bit) from each character. In some cases, the 7-bit text characters may be changed back to the 8-bit text characters by adding one bit of zero in the Most Significant Bit of each character. However, the general methodology disclosed by this invention can be applied to both cases, either 7-bit characters or 8-bit characters, by using the methods of the corresponding Text-Encoder and the corresponding Text-Decoder.

[041] Some other cellular phone network, such as CDMA (code division multiple access) cellular phone network, text character values, such as '@', defined in CDMA

SMS standards are identical to the values defined in ASCII standard. Therefore, it is not necessary to replace those characters by some other characters, as described in Step-3 of the Text-Encoder process. And at the receiver side, Step-2 of the Text-Decoder can also be ignored accordingly. If it is not necessary to transform the entire N bytes of software update data into text messages at one time, only some parts of the data can be transformed at first.

[042] 3. Method of Software Updating for Embedded Software

[043] Conventionally, the embedded software in a digital product is composed by a plurality of software functions, as shown in **Figure 2**. Normally when a function is changed, for example, the size of the function becomes bigger than before, program data in some other functions are also need to be adjusted. For example, there is a function named Function-A, and next to the Function-A there is a function named Function-B. In Function-B, there is program line to call Function-A. When Function-A is updated with a new Function-A with a bigger size than before, the offset in program command to call Function-A from Function-B becomes longer than before. Therefore, when Function-A is changed, Function-B also needs to be adjusted. This fact makes it difficult to perform software update in unit of a software function. This invention discloses a method to solve this problem. This method can make software update for one or multiple software functions, but without changing rest of functions in the embedded software.

[044] 3.1 Preparing software functions

[045] In order to support the software update functionality, every function in the embedded software can be prepared with adding a "Header-Area". The Header-Area can be created by assembly language or any other kinds of language or executable machine code. The Header-Area is used for contain program data that can be changed by software updating activities on some other functions. However, using the Header-Area, the function code itself does not need to be adjusted anymore when other functions are updated. This is because all the function calls in a function will be first directed to its corresponding Header-Area, which is an unchanged relative location to the program code in this function. Therefore, even if the real functions have been moved, offsets of command code of function calls inside the function do not need to be modified, though the offset data in the Header-Area needs to be changed so that the function calls can be directed correctly from the Header-Area. **Figure 3** shows the structure of the embedded software composed by software functions and Header-Areas. The Header-Area may also be located at the end of a function, or in the middle of a function, depending on various needs.

[046] Every function can be prepared with the following steps. Here, Function-B is used as an example.

[047] Step1: Add a "Header-Area" to the function. For example, adding a Header-Area to Function-B.

[048] Step2: find out all the functions that this function calls. For example, in Function-B there are function calls to Function-A, Function-C, and Function-D.

[049] Step3: add changes to the function so that all the functions will be directed to a place in the function Header-Area. For example, the function calls to Function-A, Function-C, and Function-D in Function-B can be directed to the Header-Area of Function-B, by creating some dummy functions for Function-A, Function-C, and Function-D in the Header Area.

[050] Step4: Modify the code in the Header-Area of the function so that all the function calls can be directed to the right places from the Header-Area. For example, in the Header-Area of Function-B, function calls to the real Function-A, Function-C, and Function-D generated, so that function calls directed to the Header-Area can be directed to the places of the right functions.

[051] 3.2 Updating software functions by re-writing embedded software

[052] When there is need to update one or multiple functions in the embedded software, using the method disclosed by this invention, only the information of the updated functions are transmitted to the digital product. Adjustment on the other functions can be performed by only changing the code in the Header-Areas of the functions. When a FLASH memory is used to contain the embedded software in the digital product, some memory buffer or spare memory space are necessary to contain some program code in order to avoid data loss during re-writing the embedded software for the new updated functions.

[053] Function updating can be performed with the following steps. Here, updating Function-A is used as an example to show the invention details.

[054] Step1: Erase the necessary area in the FLASH memory so that the new function code can be written into the FLASH memory. Note that erasing FLASH memory may need to be performed in the unit of a FLASH bank or a FLASH sector. Before erasing an area in FLASH memory, the software code in the area should be saved in a memory buffer or a spare space of the FLASH memory.

[055] Step2: Write software code into the erased area. Some of the software code can

be of the functions other than the updated function. But the updated function should also be written into the FLASH memory. For example, writing the new Function-A into FLASH memory. If the size of the new Function-A is different from that of the original Function-A, the locations of following functions can also be adjusted accordingly.

[056] Step3: Check the Header-Areas of all the functions to decide whether the offsets in program command for function call in the Header-Areas need to be adjusted. For example, the Header-Area of Function-B contains the function call to Function-A, and its corresponding offset of the function call needs to be adjusted.

[057] Step4: Adjust software code in the Header-Areas that need to be adjusted, so that the corresponding function calls can be correctly directed to the corresponding functions. For example, offset in program command the function call to Function-A in the Header-Area of Function-B should be adjusted, so that the function call can be correctly directed to the new Function-A.

[058] 3.3 Updating software functions using a reserved memory space

[059] To avoid re-writing the embedded software for updating some functions, a memory space in FLASH memory, named "Patch-Area", can be reserved beforehand, so that the new functions can be written into. It is necessary to add an Update-Processing-Routine to the beginning of each function in the embedded software. When there is a new function in the Patch-Area for updating a function, the Update-Processing-Routine at the beginning of the function will direct the execution of CPU/DSP to the new function in the Patch-Area. Designing the Update-Processing-Routine can uses the methods disclosed by a previous PCT patent application named "Embedded Software Update System" filed by Yuqing Ren on July 15, 2002 with International Application No. PCT/US 02/22412.

[060] When an update function is written into the Patch-Area, only the Update-Processing-Routine of the corresponding function needs to be modified. All the other functions including their Header-Areas can be retained without changes.

[061] When the Patch-Area is full or close to full, the entire embedded software in the FLASH memory can be re-programmed by allocating the new functions of the Patch-Area directly into program code area to replace the original functions. This involves FLASH erasing and software re-programming. The Header-Area of each function should be adjusted so that the function calls can be directly to the right places. Method of re-writing the embedded software discussed in the previous sub-section (Sec.3.2) can be also utilized here.

[062] 3.4 Software Updating in the unit of a group of functions

[063] Sections 3.1, 3.2 and 3.3 disclose methods of performing software update in the unit of a software function. This section introduces a more generic method to perform software update. That is, to perform software update in the unit of "a group of functions". Here, "a group of functions" means one or multiple functions in the embedded software. And the embedded software can be treated as the software composed by multiple groups of functions.

[064] For each group of functions, only one Header-Area is prepared to contain the program data that can be changed by software updating activities on functions other than the functions in this group. When one function in a group needs to be updated, other functions in the group, including the corresponding Header-Area should also be modified or re-programmed if necessary. One benefit of this method is that, if there are several small functions that are located together in the memory of the digital product, they can be treated as one update unit, using only one Header-Area; thus will save memory space for contain Header-Areas.

[065] Comparing with the method discussed in Section 3.1, 3.2 and 3.3, this method treats a group of functions as a unit for software update. Basically, the update methods discussed in Section 3.1, 3.2 and 3.3 can also be utilized at here, except that the unit of software update is a group of functions, instead of one function.

[066] 4. Software Update Supported by CPU/DSP hardware

[067] For supporting embedded software update/upgrade, the CPU/DSP hardware can also be designed with the consideration of update/upgrade processing. For example, CPU/DSP can have a list of buffers/registers to contain the location information of the obsolete software start addresses and their corresponding new start addresses that contains the update programs. When CPU/DSP runs to a location that is identical to an obsolete start address registered in the list, the execution of CPU/DSP will be directed to the start address of the corresponding update program. CPU/DSP may need to use some additional processing methods to support this feature.

[068] Another method is to design one or multiple CPU/DSP instructions especially for software update or upgrade. The instructions can be inserted into the embedded software, and will not perform special actions with its initial setup values, for example, when setting its parameter to "1". When there is an update program for update a software section, its setup values can be changed, for example, changing its parameter from "1" to "0". When CPU/DSP execution goes to the instruction (that has parameter "0"), this

instruction will direct the CPU/DSP execution to the start address of the update program. Some other similar methods can be considered as well.

[069] **Glossary of Abbreviations**

[070] **ASCII.** American Standards Committee for Information Interchange

[071] **CPU.** Central Processing Unit.

[072] **DSP.** Digital Signal Processor

[073] **FLASH.** A type of constantly-powered nonvolatile memory that can be erased and reprogrammed in units of memory called blocks.

[074] **MCU.** Micro Processor Unit.

[075] **PDA.** Personal Digital Assistant.

[076] **RAM.** Random Access Memory.

[077] **ROM.** Read-Only Memory.

[078] **SMS.** Short Message Service.